Testautomatisering: loont het?

Naast alle voordelen die test tooling met zich meebrengt voegen tools ook complexiteit toe: tools moeten onderhouden en geïntegreerd worden, gebruikers moeten er vaak een nieuwe taal voor leren. Allemaal acties die tijd kosten. Tijd die niet besteed kan worden aan het testen. Daarnaast is er het gevaar dat testautomatisering leidt tot het verlies van overzicht: wat test je en met welk doel.

Jeroen Peeters (ICTU): “In onze projecten willen we dat wijzigingen direct geïntegreerd getest kunnen worden. Dat kan alleen als je het proces van builden, deployen en het uitvoeren van testen automatiseert. Bij grote, complexe overheidsprojecten hanteren we een functionele testcoverage van 75%. Het uitvoeren van deze testen kan bij grote projecten al snel oplopen tot meerdere uren. Bij een groot project duurde het uitvoeren van de testen 9 uur. De test meerdere keren per dag draaien is dan geen optie meer, je krijgt pas de volgende dag het resultaat. Dat is niet acceptabel, zeker niet aan het einde van de sprint.”

Bij complexe projecten – vaak bij de overheid – worden testen vaak opgeknipt in kleinere delen om beter hanteerbaar te zijn. Peeters: “Dat opknippen is niet iets triviaals. Om tests te draaien in een opgeknipte omgeving, moet de applicatie in een bepaalde 'state' zijn, die typisch gezet wordt door de voorafgaande testen. Vaak moeten de applicaties en de tests gedeployed worden op verschillende servers. De resultaten moeten ook weer netjes geaggregeerd worden. Om dit alles op te zetten, zijn veel verschillende tools nodig, met de nodige onderlinge afhankelijkheden. Automatisering in dit soort complexe omgevingen vergt dus nogal wat handwerk vooraf – en ook het onderhoud van het proces is niet eenvoudig.”

Maarten Beks (HAYSTAQ) herkent dit: “Testautomatisering vereist een professionele omgeving, lang niet elke tool is geschikt voor elk bedrijf. Als het begint met het kiezen van de tool, zonder na te denken over de omgeving, dan gaat het fout. Het is verleidelijk om te beginnen met een tool die je kent uit een andere omgeving. Maar een tool is maar een gereedschap, je moet echt weten wat je wil bereiken en dan kan je bepalen wat voor tool je nodig hebt voor jouw doelen. Daarnaast raakt testautomatisering veel aspecten van het ontwikkel en beheerproces, het stelt ook eisen aan je IT-architectuur.”

Ook de testexperts van TMC staan vaak voor de vraag of en hoe de testen moeten worden geautomatiseerd. Benjamin Jurg ziet vaak dat onderschat wordt hoe groot de investering is om testautomatisering goed op orde te krijgen. Als op projectniveau de optimale teststrategie wordt bepaald, is het vaak effectiever te investeren in slimme, low tech oplossingen. Benjamin Jurg: “Voor onze test experts staat het bedrijfsbelang voorop, we zien vaak dat projecten nog te dynamisch zijn en de voor automatisering benodigde stabiele testbasis missen."

Jimmy Leitch van Oelan, een testbedrijf in Amsterdam dat zich al ruim elf jaar richt op testautomatisering: “Het is cruciaal om de kwaliteit van je testen goed te weten, want wat weet je nu eigenlijk over de kwaliteit van je code na de test, als je niets kan zeggen over de kwaliteit van je testen?” Een van de kansen die Oelan benut is het gebruiken van volledige kopieën van productiedata bij het testen. Hierdoor ervaren testers en developers dat defects veel eerder in de delivery pipeline worden ontdekt. Maar werken met volledige productiekopieën heeft ook een prijs. Er is veel opslagruimte nodig, *provisionen* van data kost veel tijd en het verplicht maskeren van gevoelige data is een tijdrovende klus. “Voor onze Continuous Delivery referentie architectuur zijn we daarom op zoek gegaan naar een testdata management tool die deze problemen tackelt. Met Delphix hebben we deze uiteindelijk gevonden. De tool is in staat om binnen enkele minuten volledige read/write productiekopieën met gemaskeerde data beschikbaar te stellen voor developers, test engineers en de testprocessen. Deze kopieën gebruiken netto vrijwel geen storage ook als een dataset groter is dan 50TB.”

Model Based Testen

Model Based Testen (MBT) gaat nog een een stapje verder. Naast het automatisch uitvoeren van testen, doet MBT ook de belofte van het automatisch genereren van testen. Naar MBT wordt al lange tijd onderzoek gedaan in Nederland. Het bekende succesverhaal is dat van de ontwikkeling van de software voor de Mars Rover geleid door de Nederlander Holzman. Ook Philips Healthcare heeft gekeken of MBT in het ontwikkelproces kan worden ingezet. Gernot Eggen (Philips Health Care): “Functioneel testen hebben wij inmiddels wel aardig in de vingers. De uitdaging zit in het integreren van die functie, en het testen van de combinatie van functies. Wij hebben bij Philips een aantal onderzoeken gedaan naar Model Based Testing. Op basis van dit onderzoek blijkt dat MBT op een aantal punten wat toevoegt: bijvoorbeeld

het snel kunnen leren van het gedrag van je ‘legacy component’, en het genereren van testcases voor je integratie. Bij complexe problemen ontstaat in MBT ook al snel een complex model. Fouten in een complex model zijn zeer wel denkbaar waardoor je fouten in de test cases hebt. Dit is natuurlijk niet anders dan in conventionele testautomatisering maar wel iets om rekening mee te houden door bijvoorbeeld model decompositie. Onze les was dat de systemen zich vrij lastig lieten modelleren vanwege de grote complexiteit.”

De vraag blijft relevant hoe systemen op te delen in componenten. Eggen: “Op dit moment is er een groot aanbod van modeling tools waarbij de uitdaging is om een tool te vinden dat voldoende rijk is, voldoende volwassen en een lange termijn beschikbaarheid/support biedt Eigenlijk vonden wij de tools die wij gebruikten en de leverancier in kwestie nog onvolwassen. Om ze in te zetten in het primaire proces van Philips Healthcare moeten testtools voldoen aan hoge eisen en moeten we de zekerheid hebben dat een tool jarenlang ondersteund zal worden - tenminste 20 jaar. Ook zou je in tools het gedrag van zowel het systeem als van de data goed willen combineren - en daar is nog volop ontwikkeling in. Spec Explorer was een veelbelovende tool, helaas is het onzeker of Microsoft de ondersteuning continueert.”

Een kans die Eggen ziet voor MBT is het meeleveren van een ‘testbibliotheek’ bij je component, zodat degene die het component moet integreren, daarmee eenvoudig testcases kan genereren. Eggen: “Een soort conformance test robot, waarbij je op termijn misschien automatisch verschillende modellen zou kunnen combineren. Ik verwacht daarnaast veel van het gebruik van 'machine learning' om te komen tot het selecteren van de optimale testset nu we steeds meer te maken krijgen met enorme hoeveelheden testgevallen .”

Axini is een vooraanstaande MBT toolontwikkelaars in Nederland. Machiel van der Bijl (Axini): “Bij Software Engineering maken wij complexiteit hanteerbaar door een systeem op te knippen in deelsystemen. De deelsystemen worden dan verder ontwikkeld en geïntegreerd. Met Model Based Testing zorgen wij er vervolgens voor dat we op componentniveau de kwaliteit grondig door testen, zodat we problemen kunnen uitsluiten. Het testen van het component is goed schaalbaar en goed hanteerbaar.”

In de praktijk blijkt echter dat tijdens de ontwikkeling van software het gedrag van de componenten nog volop aan verandering onderhevig is, met alle consequenties van dien. Van der Bijl: “Als je goed kijkt naar waar project-planning uitloopt, dan is dat vaak bij de problemen rondom het integreren van die subsystemen. Veel van deze problemen hebben ermee te maken dat het component niet datgene doet wat het andere component verwacht. Omdat deze verwachting vaak niet expliciet is vastgesteld, is het debuggen veel complexer en dit verlengt de ontwikkeltijd enorm. De eerste stap die wij maken in Model Based Testing is het definiëren van componenten door de bijbehorende api’s te specificeren. Verrassend vaak blijkt dan dat het helemaal niet duidelijk is wat de specificatie zou moeten zijn. Door beide kanten van de component interfaces te modelleren en te testen dwing je een goede samenwerking af. Bij het integreren kom je dan vaak alleen nog problemen tegen die ook echt op systeem niveau liggen: doen de samenwerkende componenten wel wat het systeem als geheel moet doen.”

Met andere woorden, om de complexiteit van de ontwikkeling van je systeem beheersbaar te maken is het nodig om op enig moment in de tijd het gedrag van je componenten te stabiliseren en te definiëren. Deze stap is voorwaardelijk en noodzakelijk om op component niveau goed te kunnen testen, geautomatiseerd of niet, met of zonder MBT.